Evolution as a computational engine

Rolf Backofen
Peter Clote

Ludwig-Maximilians-Universität München
Institut für Informatik
Bericht 9609

Dezember 1996

Evolution as a computational engine[†]

Rolf Backofen Peter Clote

{backofen,clote}@informatik.uni-muenchen.de
Institut für Informatik
Ludwig-Maximilians-Universität München
Oettingenstraße 67, D-80538 München, Germany.

Abstract

Given a finite set K of *lethal genes*, a starting genome x and a desired *target* genome y, is there a sequence of base insertions, deletions, and substitutions, which from x produces the desired genome y, and such that no intermediate genome contains a lethal gene. The main result of this paper is that, appropriately formulated, this problem is undecidable, even when the underlying alphabet has only 2 symbols (e.g. two of the bases A,C,G,T). We prove that asexual evolutionary systems form a universal computation engine, in the sense that for any reversible Turing machine M and input x, there is a finite set K of lethal genes, which can guide the computation, so that all intermediate genomes are prefixes of (encodings of) configurations in the computation of M. Since reversible Turing machines retain the history of their entire computation, and are essentially encoded by evolutionary systems, this suggests a encoding mechanism by which *phylogeny recapitulates ontogeny.*

Introduction

This paper studies the question of how computations can be executed in biological systems by using asexual evolution with pointwise mutations (as contrasted with sexual evolution with crossover). Our work was motivated from three directions: mathematical evolution theory, biocomputing and genetic algorithms.

[*]Research supported by NSF CCR-9408090 and US-Czech Science and Technology Grant 93-205.

Topics: formal languages, computational and structural complexity

Key words and phrases: asexual evolution, Thue system, rewrite system, reversible Turing machine.

[†]Research supported by NSF CCR-9408090 and US-Czech Science and Technology Grant 93-205.

Topics: formal languages, computational and structural complexity

Key words and phrases: asexual evolution, Thue system, rewrite system, reversible Turing machine.

Concerning evolution, it has been suggested that species in a closed environmental system are strange attractors, in the sense of mathematical dynamics. Can this remark somehow be formalized? One might try to define an evolution operator $E(x,y) = x'$, which, from the ecosystem x and environment y yields the ecosystem of the next generation. Assume that u and v are genomes (i.e. words in $\{A,C,G,T\}^*$) whose Levenshtein distance or other dissimilarity measure is small. Do both u and v have approximately the same number of descendents in each generation, when applying mutation and survival of the fittest under specified environmental conditions? Are the phylogeny trees below u and below v structurally similar? Is the evolution operator E continuous? What are the trajectories of E's iterates?

Our model of asexual evolution is admittedly a heretical biological fiction – it is known that certain genes are indispensible for the viability of an organism, but there is no evidence of lethal or harmful genes (indeed our use of the phrase 'lethal gene' is an oxymoron). Nevertheless, the simplicity of our model allows us to give a precise, mathematical formulation of the above mentioned questions which motivate our work. Our model provides us with certain interesting, highly nontrivial open problems, whose solution might help us to better understand phylogeny, genetic distance between species, and whether evolution displays chaotic behavior, in the sense of mathematical dynamics.

Concerning biocomputing, L. Adleman [1] showed how to encode graph theoretic problems using nucleotide sequences, thus illustrating how biochemical techniques could be used to solve hard combinatorial optimization problems. Our main result proves that for every computation problem, an environment can be designed which *steers* the subsequent genome populations in a manner whereby the solution of the given computation problem will appear in the progeny. Specifically, we show that *asexual evolution*, i.e. selection in an environment allowing only pointwise mutations, forms a universal computational engine, in the sense that for any algorithm A there exists a finite set K of *lethal genes* which for any input x steers the computation of A on x. All progeny genomes, not containing any lethal gene, are exactly prefixes of (encodings of) intermediate configurations of the computation of A.

Concerning genetic algorithms, in [11] P. Pudlák introduced the model of genetic Turing machine, which roughly speaking performs a computation by performing context sensitive crossover (sexual evolution), and then applying a fitness function to an exponentially large in $p(n)$ population of computation strings. Pudlák's main result was that polynomial time genetic Turing machines compute exactly polynomial space predicates, a complexity theoretic analysis which justifies the strength of genetic and evolutionary programming methods. Inde-

---

An ecosystem is a collection of species, where species is a probability distribution on a gene pool.

Levenshtein distance between sequences $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_m$ is defined as $n + m - 2\ell$, where ℓ is the length of the longest common subsequence of u, v; i.e. sequence alignment distance between u, v, with penalty 1 for a deletion or insertion and 2 for a mismatch. See chapter 9 of [12] for more on sequence alignment. See [4] for the estimated Levenshtein distance between two random sequences.

pendently, a similar result was obtained by Arora, Rabani and Vazirani [2]. By removing time and size constraints, Pudlák's construction implicitly yields a result of the form that nature, using sexual evolution with crossover and a fitness function, can simulate a universal computational device. Our main theorem, which does not follow from the work of [11, 2], shows an analogous result for asexual evolution. It is clear that one can characterize certain complexity classes by a resource bounded version of our evolutionary systems, but we have not investigated that issue here.

It should be mentioned that in [7, 8, 9] S. Yu. Maslov defined and studied various *mutation calculi*, whose rules allowed context dependent insertion, deletion and substitution as well as possible polyploidy (duplication of words). Maslov obtained results concerning the decidability and undecidability of various calculi. Our main result, though related, does not directly follow from Maslov's work, since our main proof consists of defining an appropriate finite set K of lethal genes and showing how K can indirectly act as a context for context dependent insertion, substitution and substitution. Recently, J. Dassow, V. Mitrana [5], L. Ilie, V. Mitrana [6] and G. Paun, A. Salomaa [10] have reported on various applications of formal language theory to computational biology (e.g. [5] concerns *evolutionary grammars*, which allow context sensitive insertion, deletion, and substitution rules). This work cannot be used to deduce our main results.

1 Approximate statement of result and proof overview

We wish to formalize the problem of asexual evolution in the presence of a finite set of lethal genes. For words u, v in finite alphabet Σ, write $u \subseteq v$ to mean that u is a convex subword of v, and write $u \Rrightarrow v$ if v can be obtained from u by the deletion, insertion, or substitution of one symbol. Write $u \Rrightarrow_r v$ if insertion and deletion can only occur at the end of u.

Consider the problem, given $x, y \in \Sigma^*$ and finite set $K \subset \Sigma^*$, whether there exists a sequence

$$x = z_0 \Rrightarrow z_1 \Rrightarrow \cdots \Rrightarrow z_n = y$$

for which w is not a convex subword of z_i for all $i \leq n$ and $w \in K$. This problem is trivially decidable, since there is such a sequence if and only if no lethal gene $w \in K$ is a convex subword of either x or y.

There are various manners to remedy this trivial formulation of the problem. In a biological setting, special proteins bind to ends of chromosomes to help prevent unravelling, so it seems reasonable to allow special *end markers* which are not part of the alphabet Σ. This is incorporated in the following definition.

Definition 3.1 *Let Σ be a finite alphabet, consisting of at least 2 symbols, and*

Personal communication, Jan Krajíček.

$\$, ¢ \notin \Sigma$. *Let* $x, y \in \$\Sigma^*¢$ *and* K *be a finite subset of* $(\Sigma \cup \{\$, ¢\})^*$. *Write*

$$x \Rrightarrow^*_{\Sigma,K} y$$

to denote that for some natural number n

$$x = x_0 \Rrightarrow x_1 \Rrightarrow \cdots \Rrightarrow x_n = y$$

and for all $i \leq n$ *and* $w \in K$, $w \not\subseteq \$x_i¢$. *As well, write* $x \Rrightarrow^*_{\Sigma,K,r} y$, *when insertions and deletions can only occur at the right extremity. A tuple* $E = (\Sigma, K)$ *is called an* *evolutionary system.*

Definition 4.1 (Future Genome Problem) *Fix* Σ *be a finite alphabet,* $|\Sigma| \geq 2$, *and* $\$, ¢ \notin \Sigma$. *Given* $x, y \in \$\Sigma^*¢$, *and finite subset* $K \subseteq (\Sigma \cup \{\$, ¢\})^*$, *does* $x \Rrightarrow^*_{\Sigma,K} y$*?*

A similar problem is the *Future Gene Problem.* Given $K \subseteq (\Sigma \cup \{\$, ¢\})^*$, $x \in \$\Sigma^*¢$ and $y \in \Sigma^*$, does $x \Rrightarrow^*_{\Sigma,K} z$ for some z with $y \subseteq z$? The undecidability of this problem follows from the undecidability of the future genome problem (by introducing two additional end markers).

Definition 4.2 (Strong Future Genome Problem) *Fix* Σ *be a finite alphabet,* $|\Sigma| \geq 2$, *and* $\$, ¢ \notin \Sigma$, *and finite subset* $K \subseteq (\Sigma \cup \{\$, ¢\})^*$. *Given* $x, y \in \$\Sigma^*¢$, *does* $x \Rrightarrow^*_{\Sigma,K} y$*?*

Theorem 4.3 *For every* Σ *with* $|\Sigma| \geq 2$, *there exists finite* K *for which the Strong Future Genome Problem is undecidable.*

Theorem 4.3 follows as a corollary of the following result, which is only approximately stated.

> Let $M = (Q, \Gamma, \delta, s)$ be a 1-tape Turing machine. There exists a finite set $K \subset \{0, 1, \$, ¢\}^*$ such that for all inputs $w \in \Gamma^*$ to M, if $x \in \$\{0,1\}^*¢$ denotes a natural encoding of w, then $\{y \in \$\{0,1\}^*¢ : x \Rrightarrow^*_{\Sigma,K,r} y\}$ consists only of prefixes of encodings of configurations in the computation of M on w.

In this section, we sketch the main ideas for the proof of the previous statement. By Bennett [3], every 1-tape TM is equivalent to a 3-tape reversible TM. Encode the reversible TM by an equivalent rewriting system, where rules are of the form

$$\begin{array}{ll} uaqbv \Rightarrow uq'abv \mid uaq'b'v \mid uabq'v & \\ [\#q\#v \Rightarrow [\#q'v & \text{(move right)} \\ u\#q\#] \Rightarrow uq'\#] & \text{(move left)} \\ uqa] \Rightarrow uaq'\#] & \text{(move right).} \end{array}$$

The first rule handles ordinary transitions, and the last 3 rules handle transitions where the TM head is at an extremity, and ensure that ## is neither a prefix of u nor a suffix of v, where uqv is a configuration. Clearly, deletions and insertions are only made at the right. Since we have a reversible Turing machine, the resulting rewrite system is a Thue-system.

Next, modify the previous rewrite system, so that rules change only one symbol (rather than a string of symbols), the idea being that $uabv \Rightarrow ucdv$ can be simulated by the following:

$$\begin{array}{rcl} uabv & \Rightarrow & u\bar{a}bv \\ & \Rightarrow & u\bar{a}\bar{b}v \\ & \Rightarrow & uc\bar{b}v \\ & \Rightarrow & ucdv \end{array}$$

Next, define an equivalent rewrite system in the alphabet $\{0,1\}$ by encoding the symbols of the previous system's alphabet in unary. In other words, if the previous rewrite alphabet has m symbols, then encode the i-th symbol by $11100^{i-1}10^{m-i}0111$. Here the initial (resp. terminal) string 1110 (resp. 0111) acts as a buffer to allow concatenation of encodings of symbols. The rules of the rewrite system are encodings of the rules of the previous system. Unlike the previous system, this rewrite system no longer has rules which change only one symbol.

Next, define an equivalent rewrite system in alphabet $\{0,1\}$ whose rules change only one symbol by replacing each previous rule

$$11100^{i-1}10^{m-i}0111 \Rightarrow 11100^{j-1}10^{m-j}0111$$

(this rule transforms the encoding of the i-th symbol to the encoding of the j-th symbol, where for specificity we assume $i < j$) by the two following rules

$$\begin{array}{rcl} 11100^{i-1}10^{m-i}0111 & \Rightarrow & 11100^{i-1}10^{j-i-1}10^{m-j}0111 \\ & \Rightarrow & 11100^{j-1}10^{m-j}0111. \end{array}$$

Finally, one essentially defines K to be those strings of the form $1110w0111$ where $|w|_1 \notin \{1,2\}$, or $|w|_1 = 2$ and which don't correspond to a transformation of the type illustrated above. Additional details must be applied to ensure insertion and deletion only at the right. This completes the sketch of proof of Theorem 4.3.

2 Thue Systems and Evolutionary Systems

For the remainder of this paper, Σ will denote a finite alphabet with $|\Sigma| \geq 2$, and $\$, ¢ \notin \Sigma$. For words $x, y \in \Sigma^*$, write xy to denote the concatenation of x, y. The empty word is denoted by λ. The word x is a prefix [resp. suffix] of the word y if $y = xu$ [resp. $y = ux$] for some word u. The word x is a convex subword of word y, denoted by $x \subseteq y$, if $y = uxv$ for words u, v. The length of word x is denoted by $|x|$; e.g. $|x_1 \cdots x_n| = n$, if $x_i \in \Sigma$. If $a \in \Sigma$ and $x \in \Sigma^*$, then $|x|_a$ is the number of occurrences of symbol a in x.

Given a word $u \in \Sigma$ and a number $1 \leq i \leq |u|$, we define u_i to be the symbol at position i of u. For $1 \leq i \leq j \leq |u|$, we write u_i^j to denote the substring of u between positions i and j (including i, j). For convenience, we define $u_i^j := u_1^j$ for $i < 1$, $u_i^j := u_i^{|u|}$ for $j > |u|$ and u_i^j to be the empty string for $j < 1$, $i > |u|$ or $i > j$.

Definition 6.1 *A semi-Thue System over a finite alphabet Σ is a finite subset G of $\Sigma^* \times \Sigma^*$. A Thue System over Σ is a finite binary, symmetric relation $G \subset \Sigma^* \times \Sigma^*$. We write a Thue System G as a set of unordered pairs (i.e. sets with two elements, where $\{u, v\} \in G$ means that both $(u, v) \in G$ and $(v, u) \in G$). An unordered pair $\{u, v\} \in G$ is called a rule. We write*

$$x \Rightarrow_{(u,v)} y$$

if $x = wuw'$ and $y = wvw'$, and we define $\Rightarrow_{\{u,v\}}$ as the symmetric closure of $\Rightarrow_{(u,v)}$. We write $x \Rightarrow_G y$ if $x \Rightarrow_{(u,v)} y$ for some $(u, v) \in G$. With $\Rightarrow_G^*$ we denote the reflexive and transitive closure of $\Rightarrow_G$.

A derivation in G (or G-derivation) is a sequence

$$x_0 \Rightarrow_{r_1} x_1, \cdots, x_{n-1} \Rightarrow_{r_n} x_n$$

such that $\{r_1, \ldots, r_n\} \subseteq G$.

2.1 Reducing Thue Systems to Substitution Systems

Definition 6.2 (Substitution System) *A substitution system P is a triple (Σ, δ, R) consisting of an alphabet Σ, a designated element $\delta \in \Sigma$ and a finite set of rules $R \subseteq \Sigma^* \times \Sigma^*$ of the form $\{u, v\}$ with $|u| = |v|$. The associated Thue-system G_P is a Thue system over $\Sigma \cup \{\$, ¢\}$ defined by*

$$G_P := R \cup \{\{¢, \delta¢\}\}.$$

We define $\Rightarrow_P$ to be $\Rightarrow_{G_P}$. For every $r = \{u, v\} \in R$ we define the length of r (written $\mathrm{len}(r)$) to be $|u|$. A substitution system is called simple if $\{u, v\} \in R$ implies that u and v differ in at most one symbol.

Note that given a derivation

$$x_0 \Rightarrow_{r_1} x_1 \Rightarrow_{r_2} \cdots x_{n-1} \Rightarrow_{r_n} x_n$$

in P, then not all of the r_i's are elements of R. Rather, some of the r_i's may be the deletion/insertion rule $\{¢, \delta¢\}$.

Lemma 6.3 *Every Thue System G over Σ can be simulated by a substitution system $P = (\Sigma \cup \{\delta\}, \delta, R)$ such that for all $x, y \in \Sigma^*$, $x \Rightarrow_G^* y$ if and only if $\$x¢ \Rightarrow_P^* \$y¢$.*

Proof. Let G be a fixed Thue System over Σ. Construct a substitution system P as follows. We write δ^k as an abbreviation for the word of length k consisting only of δ's. The set R is defined as

$$\begin{array}{rcl} R & := & R_{trans}(G) \cup R_{exch}(G) \\ R_{trans}(G) & := & \{\{u, v\delta^k\} \mid \{u, v\} \in G \wedge |u| \geq |v| \wedge k = |u| - |v|\} \\ R_{exch}(G) & := & \{\{\delta\sigma, \sigma\delta\} \mid \sigma \in \Sigma\} \end{array} \quad (1)$$

The rules in (1) are called exchange rules. Note well that for different rules r, r', it may be that $\text{len}(r) \neq \text{len}(r')$.

Then for $x, y \in \Sigma^*$, we have $x \Rightarrow_G^* y$ if and only if $\$x¢ \Rightarrow_P^* \$y¢$:

- $\rightarrow$ We simulate a rule application $z \Rightarrow_{(u,v)} z'$ with $(u, v) \in G$ and $|u| \geq |v|$ by a sequence of rule applications in P of the form

  $$\$z¢ = z_0 \Rightarrow_{r_1} z_1 \ldots z_{n-1} \Rightarrow_{r_n} z_n = \$z'¢,$$

  where r_1 is the rule $(u, v\delta^k) \in R_{trans}(G)$ with $k = |u| - |v|$, $r_2, \ldots, r_{n-k}$ are applications of the exchange rules in (1) transporting the δ's introduced by the application of r_1 to the right end, and $r_{n-k+1} \ldots r_n$ are applications of the right deletion/insertion rule $\{¢, \delta¢\}$ successively deleting the introduced δs. A rule application $z \Rightarrow_{(v,u)} z'$ with $(v, u) \in G$ and $|v| < |u|$ is simulated analogously.

- $\leftarrow$ Let

  $$\$x¢ = z_0 \Rightarrow_{r_1} z_1 \ldots z_{n-1} \Rightarrow_{r_n} z_n = \$y¢$$

  be a derivation in P such that $x, y \in \Sigma^*$. For every word $u \in (\Sigma \cup \{\delta, \$, ¢\})^*$, let $u/\{\delta, \$, ¢\}$ be the word in Σ obtained by deleting all occurrences of $\delta, \$$ and $¢$ in u. Now for every rule $r = (u, v\delta^k) \in R_{trans}(G)$ and $x \Rightarrow_r y$ we have $x/\{\delta, \$, ¢\} \Rightarrow_{\{u,v\}} y/\{\delta, \$, ¢\}$. For every $r \notin R_{trans}(G)$ and $x \Rightarrow_r y$ we have $x/\{\delta, \$, ¢\} = y/\{\delta, \$, ¢\}$. Hence, we can transform every derivation sequence in P into a valid sequence in G by deleting all δ's and all exchange and deletion/insertion rules, and by transforming every rule $R_{trans}(G)$ into a corresponding G rule.

□

¿From undecidability of Thue systems, one has the following.

Proposition 7.1 *There is a substitution system P such that it is undecidable, given words x, y, whether $x \Rightarrow_P^* y$.*

In the following sections, we show that one can simulate arbitrary substitution systems P by evolutionary systems. The simulation is done by performing the following steps:

1. A substitution system $P = (\Sigma, \delta, R)$ is translated into a simple substitution system $P' = (\Sigma', \delta, R')$. The translation guarantees that every derivation in P can be simulated in P', and specific derivations in P' satisfying a given order for rule applications can be translated back into P.
2. The simple substitution system P' is then simulated by a restricted evolutionary system $E_{restr} = (\Sigma', \delta, R_S, K)$ (i.e., an evolutionary system where not all substitution rules $\{\sigma, \sigma'\}$ are allowed but only those contained in a given set R_S, and where only the symbol δ may be deleted). In this system, the application of a rule in R' is simulated by the substitution of the one symbol that is changed. The lethal genes will guarantee that
   (a) the substitution of a symbol is only allowed if it is in the right context
   (b) the substitutions can be reordered in a way such that the corresponding rule applications in P' can be translated back into P.
3. E_{restr} is then translated into an evolutionary system E (with alphabet $\{0, 1\}$).

As an example for the simulation indicated above in step 1, suppose that R contains only the rule

$$\{ab, ef\}. \tag{2}$$

Then this rule is simulated in P' by a sequence of rules of the form

$$\begin{aligned} ab &\Rightarrow \sigma^{r,1}b \\ &\Rightarrow \sigma^{r,1}\sigma^{r,2} \\ &\Rightarrow e\sigma^{r,2} \\ &\Rightarrow ef. \end{aligned}$$

Here, the $\sigma^{r,i}$ are new symbols, namely for every rule and every position in the left (or right side) of the given rule in R.

This special form for the rules can then be translated into a restricted evolutionary system as described in step 2. Here, we allow the substitution of a symbol σ which occurs in the position i of a rule r by the symbol $\sigma^{r,i}$. For our example (2), the substitutions used are

$$\{a, \sigma^{r,1}\},\ \{e, \sigma^{r,1}\},\ \{b, \sigma^{r,2}\},\ \{f, \sigma^{r,2}\}.$$

2.2 Simulating Substitution Systems by Simple Substitution Systems

For the simulation of P we define a simple substitution system $P' = (\Sigma', \delta, R')$ with

$$\begin{aligned}
\Sigma' &:= \Sigma \cup \Sigma_{new} \\
\Sigma_{new} &:= \{\sigma^{r,i} \mid r = \{u, v\} \in R, 1 \leq i \leq |u|\} \\
R' &:= \{r \in R \mid \mathrm{len}(r) = 1\} \cup \bigcup_{r \in R, \mathrm{len}(r) > 1} \mathrm{trans}(r)
\end{aligned}$$

where the $\sigma^{r,i}$ are new symbols not occurring in Σ, and $\mathrm{trans}(r)$ is given as follows. For $r = \{u, v\}$ with $|u| > 1$ we define $\mathrm{trans}(r)$ by

$$\mathrm{trans}(r) \quad := \quad \mathrm{trans}_1(r) \cup \bigcup_{1<i<|u|} \mathrm{trans}_i(r) \ \cup \mathrm{trans}_{|u|}(r)$$

where

$$\begin{aligned}
\mathrm{trans}_1(r) &:= \left\{ \left\{ \begin{array}{c} \boldsymbol{u_1} u_2 \ldots u_{|u|}, \\ \boldsymbol{\sigma^{r,1}} u_2 \ldots u_{|u|} \end{array} \right\}, \left\{ \begin{array}{c} \sigma^{r,1}\ \boldsymbol{u_2}\ u_3 \ldots u_{|u|}, \\ \sigma^{r,1} \boldsymbol{\sigma^{r,2}} u_3 \ldots u_{|u|} \end{array} \right\} \right\} \\
\forall 1 < i < |u| : & \\
\mathrm{trans}_i(r) &:= \left\{ \left\{ \begin{array}{l} v_1 \ldots v_{i-2} \sigma^{r,i-1} \sigma^{r,i}\ \boldsymbol{u_{i+1}}\ u_{i+2} \ldots u_{|u|}, \\ v_1 \ldots v_{i-2} \sigma^{r,i-1} \sigma^{r,i} \boldsymbol{\sigma^{r,i+1}} u_{i+2} \ldots u_{|u|} \end{array} \right\} \right\} \\
&\cup \left\{ \left\{ \begin{array}{l} v_1 \ldots v_{i-2} \boldsymbol{\sigma^{r,i-1}} \sigma^{r,i} \sigma^{r,i+1} u_{i+2} \ldots u_{|u|}, \\ v_1 \ldots v_{i-2}\ \boldsymbol{v_{i-1}}\ \sigma^{r,i} \sigma^{r,i+1} u_{i+2} \ldots u_{|u|} \end{array} \right\} \right\} \\
\mathrm{trans}_{|u|}(r) &:= \left\{ \left\{ \begin{array}{l} v_1 \ldots v_{|u|-2} \boldsymbol{\sigma^{r,|u|-1}} \sigma^{r,|u|}, \\ v_1 \ldots v_{|u|-2}\ \boldsymbol{v_{|u|-1}}\ \sigma^{r,|u|} \end{array} \right\}, \left\{ \begin{array}{l} v_1 \ldots v_{|u|-1} \boldsymbol{\sigma^{r,|u|}}, \\ v_1 \ldots v_{|u|-1} \boldsymbol{v_{|u|}} \end{array} \right\} \right\}
\end{aligned}$$

For every r, we define $\mathrm{tr}_{r,1} \cdots \mathrm{tr}_{r,m_r}$ to be the enumeration of the rules in $\mathrm{trans}(r)$ in the order they are defined above. Note that this encoding has the property that every word used in a rule $\mathrm{trans}_i(r)$ with $1 < i < |u|$ contains at least two of the new symbols. It is easy to see that a rule application $x \Rightarrow_r y$ in P can be simulated by the P'-derivation

$$x \Rightarrow_{\mathrm{tr}_{r,1}} \ldots \Rightarrow_{\mathrm{tr}_{r,m_r}} y.$$

Clearly, there is a simpler encoding which has the property. For example, consider the rule

$$r = \{abc, efg\}.$$

Then we could simulate an application of this rule using the derivation

$$abc \Rightarrow \sigma^{r,1}bc \Rightarrow \sigma^{r,1}\sigma^{r,2}c \Rightarrow e\sigma^{r,2}c \Rightarrow e\sigma^{r,2}\sigma^{r,3} \Rightarrow ef\sigma^{r,3} \Rightarrow efg \qquad (3)$$

for better readability, we write a rule $\{\ u_1 \ldots u_{j-1} \sigma u_{j+1} \ldots u_{|u|}\ ,\ u_1 \ldots u_{j-1} \tau u_{j+1} \ldots u_{|u|}\ \}$ as $\left\{ \begin{array}{l} u_1 \ldots u_{j-1} \boldsymbol{\sigma} u_{j+1} \ldots u_{|u|}, \\ u_1 \ldots u_{j-1} \boldsymbol{\tau} u_{j+1} \ldots u_{|u|} \end{array} \right\}$

We use the more complex version since the simple substitution system $P'_{abc,efg}$ corresponding to (3) cannot be correctly simulated by a restricted evolutionary system. The reason is that we need the rule $\{b, \sigma^{r,2}\}$ for simulating $P'_{abc,efg}$. Hence, a possible transformation in the restricted evolutionary system is

$$ebc \Rightarrow_{\{b,\sigma^{r,2}\}} e\sigma^{r,2}c$$

which is invalid, but cannot be excluded using lethal genes.

Next, we define the P'-derivations that can be translated back into P-derivations. A rule application $x \Rightarrow_r y$ in P' is called <u>initial</u> (resp. <u>final</u>) if $x = uvw$ and $y = uv'w$ such that $v \in (\Sigma \cup \{\$, ¢\})^*$ (resp. $v' \in (\Sigma \cup \{\$, ¢\})^*$). We say that a rule application $x \Rightarrow_r y$ is <u>interior</u> if it is neither initial nor final, i.e. v and v' contain an element of Σ_{new}. Note that the deletion/insertion rule is a rule whose application is both initial and final at the same time.

Given two consecutive rule applications

$$x \Rightarrow_r y \Rightarrow_{r'} z$$

in P', we say that the applications of r and r' are <u>P-consecutive</u> iff the following conditions hold:

1. if the application of r is final, then the application of r' is initial,
2. if the application of r is not final, then $x = uvw \wedge y = uv'w \wedge z = uv''w$ such that
$$r = \{v, v'\} \wedge r' = \{v', v''\}.$$

Note that in the second case it is allowed that r equals r'. Then we have $v = v''$ and hence $x = z$, i.e., the second application of r is just undoing the effects of the first application of r. Furthermore, note that v, v', v'' may be words in Σ^*, rather than symbols in Σ. We say that a derivation in P' is <u>pure</u> if every two consecutive rule applications are P-consecutive.

Proposition 10.1 *Let P and P' be given as described. Then every derivation in P can be translated into an equivalent derivation in P' which is pure. Conversely, every pure derivation in P' where both the initial word and the final word is in $\$\Sigma^*¢$ (i.e., are words over the alphabet of P) can be translated into an equivalent derivation in P.*

Hence, our next step should be to show that one can "reorder" every derivation in P' into a pure derivation. But this is not always possible. E.g., consider the substitution system P such that R contains the rules

$$r = \{ab, cd\} \quad \text{and} \quad r' = \{xc, yc\}.$$

Then

$$
\begin{array}{rcllll}
\mathrm{trans}(r) & := & \{ & \{ab, \sigma^{r,1}b\} & \{\sigma^{r,1}b, \sigma^{r,1}\sigma^{r,2}\} & \\
& & & \{\sigma^{r,1}\sigma^{r,2}, c\sigma^{r,2}\} & \{c\sigma^{r,2}, cd\} & \} \\
\mathrm{trans}(r') & := & \{ & \{xc, \sigma^{r',1}c\} & \{\sigma^{r',1}c, \sigma^{r',1}\sigma^{r',2}\} & \\
& & & \{\sigma^{r',1}\sigma^{r',2}, y\sigma^{r',2}\} & \{y\sigma^{r',2}, yc\} & \}.
\end{array}
$$

Hence, one possible P'-derivation is

$$
\begin{array}{rllll}
xab & \Rightarrow_{P'} & x\sigma^{r,1}b & \Rightarrow_{P'} & x\sigma^{r,1}\sigma^{r,2} \\
& \Rightarrow_{P'} & xc\sigma^{r,2} & \Rightarrow_{P'} & \sigma^{r',1}c\sigma^{r,2} \\
& \Rightarrow_{P'} & \sigma^{r',1}\sigma^{r',2}\sigma^{r,2} & \Rightarrow_{P'} & y\sigma^{r',2}\sigma^{r,2} \\
& \Rightarrow_{P'} & yc\sigma^{r,2} & \Rightarrow_{P'} & y\sigma^{r,1}\sigma^{r,2} \\
& \Rightarrow_{P'} & y\sigma^{r,1}b & \Rightarrow_{P'} & yab.
\end{array}
$$

Now this can not be reordered into a pure derivation. The reason is just that we start in the second row with the simulation of rule r' without completing the simulation of rule r. Later in the fourth row, we undo the simulation of the rule r. Hence, there is no direct translation of this derivation into a P-derivation with two rule applications. Nevertheless, there is an equivalent P-derivation, namely

$$xab \Rightarrow_r xcd \Rightarrow_{r'} ycd \Rightarrow_r yab.$$

Hence, we conjecture that it is possible to "complete" every P'-derivation such that the completed derivation can be reordered into a pure derivation. However, we will not prove this, since it is easier to simulate P' in a restricted evolutionary system and then ensure using lethal genes that such derivations do not occur.

2.3 Simulating Substitution Systems by Restricted Evolutionary Systems

Definition 11.1 (Restricted Evolutionary System) *Let $\mathcal{P}_2(\Sigma)$ denote the set of all 2-element subsets of Σ. A restricted evolutionary system $E_{restr} = (\Sigma, \delta, R_S, K)$ is an evolutionary system, where $R_S \subseteq \mathcal{P}_2(\Sigma)$ and $\delta \in \Sigma$ is the only symbol that may be inserted or deleted. The associated Thue-system over the alphabet $\Sigma \cup \{\$, ¢\}$ is defined as*

$$G_{E_{restr}} \quad := \quad R_S \cup \{\{¢, \sigma ¢\} \mid \sigma \in \Sigma\}$$

We say that a derivation

$$x_0 \Rightarrow_{G_{E_{restr}}} \ldots \Rightarrow_{G_{E_{restr}}} x_n$$

is valid in E_{restr} (or a valid E_{restr}-derivation) if $\forall i \in 0..n, \forall w \in K : w \not\subseteq x_i$.

In private correspondence, P. Pudlák pointed out a similar example leading to this problem.

We simulate P' by a restricted evolutionary system $E_{restr} = (\Sigma', \delta, R_S, K)$, where we let

$$R_S \quad := \quad \{\{u_i, \sigma^{r,i}\} \mid 1 \leq i \leq |u| \wedge r = \{u, v\} \in R\}.$$

The set of lethal genes K will be defined later. Note that P' and E_{restr} use the same alphabet Σ', and the same symbol δ that may be deleted. We say that a substitution $\{\sigma, \sigma^{r,i}\}$ simulates a rule $\{u, v\}$ if $u \Rightarrow_{\{\sigma,\sigma^{r,i}\}} v$.

Proposition 12.1 *Every substitution in R_S simulates at most two rules. Furthermore, a substitution $\{\sigma, \sigma^{r,i}\}$ simulates two rules if and only if $r = \{u, v\}$ and $u_i = v_i$.*

Consider our example (2) where we have

$$\begin{array}{rcl}
R & := & \{\ \{ab, ef\}\ \} \\
R' & := & \{\ \{ab, \sigma^{r,1}b\},\ \{\sigma^{r,1}b, \sigma^{r,1}\sigma^{r,2}\},\ \{\sigma^{r,1}\sigma^{r,2}, e\sigma^{r,2}\},\ \{e\sigma^{r,2}, ef\}\ \} \\
R_S & := & \{\ \{a, \sigma^{r,1}\},\ \{e, \sigma^{r,1}\},\ \{b, \sigma^{r,2}\},\ \{f, \sigma^{r,2}\}\ \}.
\end{array}$$

Then every rule in R_S simulates exactly one rule in R' (e.g., $\{a, \sigma^{r,1}\}$ simulates $\{ab, \sigma^{r,1}b\}$ etc.). If we had started with a set

$$R \quad := \quad \{\ \{ab, af\}\ \}$$

then the corresponding sets R' and R_S would be

$$\begin{array}{rcl}
R' & := & \{\ \{ab, \sigma^{r,1}b\},\ \{\sigma^{r,1}b, \sigma^{r,1}\sigma^{r,2}\},\ \{\sigma^{r,1}\sigma^{r,2}, a\sigma^{r,2}\},\ \{a\sigma^{r,2}, af\}\ \} \\
R_S & := & \{\ \{a, \sigma^{r,1}\}, \{b, \sigma^{r,2}\},\ \{f, \sigma^{r,2}\}\ \}.
\end{array}$$

In this setting, the rule $\{a, \sigma^{r,1}\}$ would simulate the two rules $\{ab, \sigma^{r,1}b\}$ and $\{\sigma^{r,1}\sigma^{r,2}, a\sigma^{r,2}\}$ in R'.

Obviously, we can simulate every derivation in P' by the corresponding sequence of substitution, deletions or insertions. For the back translation, we have to ensure using a set K_{con} of lethal genes that every substitution corresponds to the application of one of the rules in some $\text{trans}(r)$.

A word is called admissible for a symbol $\sigma^{r,i}$ if it is an element of

$$\text{adm}(\sigma^{r,i}) \quad := \quad \{w \in \Sigma'^*\sigma^{r,i}\Sigma'^* \mid \exists w' : \{w, w'\} \in \text{trans}(r)\}.$$

We define $\text{adm}(r) := \bigcup_{1 \leq i \leq \text{len}(r)} \text{adm}(\sigma^{r,i})$. Note that if w is admissible for $\sigma^{r,i}$, then it has length $\text{len}(r)$.

Given a word w and a symbol $\sigma^{r,i}$ with $r = \{u, v\}$ such that $\sigma^{r,i}$ occurs at position k in w (i.e., $\sigma^{r,i} = w_k$)), the (w, k)-context of $\sigma^{r,i}$ is defined as $w_{k-i+1}^{k+\text{len}(r)-i}$.

Now the first idea is to include in K_{con} for every rule r and for every $1 \leq i \leq \text{len}(r)$ all words in $\Sigma'^{\text{len}(r)} \backslash \text{adm}(\sigma^{r,i})$. But this is not enough since we must forbid a symbol $\sigma^{r,i}$ to occur too close to the beginning or end of a word produced by the evolutionary system. Thus, we define

$$\begin{aligned}
K_{con}^{r,i} &:= \{w\sigma^{r,i}w' \mid && w, w' \in (\Sigma' \cup \{\$, ¢\})^* \\
& && \wedge\ |w| \leq i-1 \wedge |w'| \leq \mathrm{len}(r) - i \\
& && \wedge\ \forall z, z' \in \Sigma'^* : zw\sigma^{r,i}w'z' \notin \mathrm{adm}(\sigma^{r,i})\} \\
K_{con} &:= \bigcup_{r \in R} \bigcup_{1 \leq i \leq \mathrm{len}(r)} K_{con}^{r,i}.
\end{aligned}$$

Note that K_{con} contains more lethal genes than needed, but this simplifies the proofs.

Proposition 13.1 *Let $w \in \$\Sigma'^*¢$. Then w is excluded by the lethal genes K_{con} if and only if it contains a symbol $\sigma^{r,i}$ at some position k such that the (w,k)-context of $\sigma^{r,i}$ is not admissible for $\sigma^{r,i}$.*

Proof. For every word w and every $\sigma^{r,i}$ at position k of w, the (w,k)-context z of $\sigma^{r,i}$ is admissible if and only if every substring of z containing $\sigma^{r,i}$ is not in K_{con}. □

Lemma 13.2 *Let $x \in \$\Sigma'^*¢$, and let $x \Rightarrow_s y$ with s being the substitution $\{\sigma, \sigma^{r,i}\} \in R_S$. If neither x nor y is excluded by the lethal genes K_{con}, then there is a unique $r' \in \mathrm{trans}(r)$ in R' simulated by $\{\sigma, \sigma^{r,i}\}$ such that $x \Rightarrow_{r'} y$.*

Proof. Let x, y be given as described, let $s = \{\sigma, \sigma^{r,i}\}$, where $r = \{u, v\}$ and $1 \leq i \leq \mathrm{len}(r)$. Let k be the position in x where the substitution has taken place. Let

$$z = x_{k-i+1}^{k+\mathrm{len}(r)-i} \quad \text{and} \quad z' = y_{k-i+1}^{k+\mathrm{len}(r)-i}$$

Since either z or z' contains the symbol $\sigma^{r,i}$, we know by Proposition 13.1 that one of z, z' is an element of $\mathrm{adm}(r)$. In the following we consider the case that z' contains $\sigma^{r,i}$. An analogous argument applies if z contains $\sigma^{r,i}$.

If $z' \in \mathrm{adm}(r)$, then z must also be in $\mathrm{adm}(r)$: if z contains a symbol from Σ_{new} say $\sigma^{r',j}$, then r' must equal r since otherwise z' would contain two symbols $\sigma^{r,i}$ and $\sigma^{r',j}$ with $r \neq r'$, which would contradict our assumption $z' \in \mathrm{adm}(r)$; if z does not contain a symbol from Σ_{new} (i.e., $z \in \Sigma^*$), then i must be 1 or $\mathrm{len}(r)$ since otherwise we could again conclude $z' \notin \mathrm{adm}(r)$.

Hence, we know that there are words w, w' such that $\{z, w\} \in \mathrm{trans}(r)$ and $\{z', w'\} \in \mathrm{trans}(r)$. If we had $\{z, z'\} \notin \mathrm{trans}(r)$, then by the definition of $\mathrm{trans}(r)$ the Hamming distance of z and z' would be either 0 or greater than 1 since z' contains the symbol $\sigma^{r,i}$. This would contradict our assumption that z' is obtained from z by a single substitution. Hence, $\{z, z'\}$ is a rule in $\mathrm{trans}(r)$ that is simulated by $\{\sigma, \sigma^{r,i}\}$ with $x \Rightarrow_{\{z,z'\}} y$. □

Corollary 13.3 *Every derivation valid in the restricted evolutionary system* $E'_{restr} = (\Sigma', \delta, R_S, K_{con})$ *has an equivalent derivation in* P' *and vice versa.*

Definition 13.4 *Given a derivation in* P', *we denote with* $\mathrm{sim}_{evol}(D)$ *the equivalent derivation that is valid in* $E'_{restr} = (\Sigma', \delta, R_S, K_{con})$. *Conversely, given a derivation* D *valid in* E'_{restr}, *we denote with* $\mathrm{sim}_{subst}(D)$ *the unique derivation in* P' *which is equivalent to* D.

The final observation is that we can re-order every derivation D in the evolutionary system E'_{restr} such that $\mathrm{sim}_{subst}(D)$ is a pure P'-derivation. For this purpose, we need another set of lethal genes. These lethal genes will guarantee that different applications of substitution interact within the corresponding context if and only if they are P-consecutive rule applications. Let

$$K_{inter} \quad := \quad \{\sigma^{r,i} w \sigma^{r',j} \mid w \in \Sigma'^* \wedge 1 \leq |w| \leq \mathrm{len}(r) - i + j - 1\}$$

In the remaining part of the simulation, we set $E_{restr} = (\Sigma', \delta, R_S, K_{con} \cup K_{inter})$.

Proposition 14.1 *Let* D *be a pure* P'*-derivation. Then* $\mathrm{sim}_{evol}(D)$ *is a valid derivation in* E_{restr}.

Proof. We have only to show that in such derivations, no lethal gene in K_{inter} can be applied. But this is trivial since if ever two symbols $\sigma^{r,i}$ and $\sigma^{r',j}$ occur in a word in the derivation $\mathrm{sim}_{evol}(D)$, they are consecutive. However K_{inter} contains no word having this property. $\square$

Lemma 14.2 *Let* D *be a valid* E_{restr}*-derivation with*

$$\mathrm{sim}_{subst}(D) \quad = \quad x \Rightarrow_{r_1} y \Rightarrow_{r_2} z$$

such that the applications of r_1 *and* r_2 *in* $\mathrm{sim}_{subst}(D)$ *are not* P*-consecutive. Then there is some* y' *and an valid* E_{restr}*-derivation* D' *such that*

$$\mathrm{sim}_{subst}(D') \quad = \quad x \Rightarrow_{r_2} y' \Rightarrow_{r_1} z.$$

Proof (sketch). The key observation is that since the applications of r_1 and r_2 are not P-consecutive, K_{inter} guarantees that either

$$x = usvtw \wedge y = us'vtw \wedge z = us'vt'w$$

such that $r_1 = \{s, s'\}$ and $r_2 = \{t, t'\}$, or

$$x = usvtw \wedge y = usvt'w \wedge z = us'vt'w$$

such that $r_2 = \{t, t'\}$ and $r_2 = \{s, s'\}$. $\square$

Lemma 14.3 (Reordering) *Let $x, y \in \$\Sigma^*\cent$ and D be a derivation valid in E_{restr}, and let* $\mathrm{sim}_{subst}(D)$ *be*

$$x = x_0 \Rightarrow_{r_1} x_1 \ldots x_{n-1} \Rightarrow_{r_n} x_n = y.$$

Then there is a permutation π of $\{1, \ldots, n\}$ such that

$$x = x'_0 \Rightarrow_{r_{\pi(1)}} x'_1 \cdots x'_{n-1} \Rightarrow_{\pi(r_n)} x'_n = y$$

is a pure P'-derivation.

Lemma 15.1 *For every derivation in P there is an equivalent derivation in E_{restr} and vice versa.*

Proof. This follows from Propositions 10.1 and 14.1, and Lemma 14.3. □

2.4 Simulating Restricted Evolutionary Systems

Finally, we simulate the restricted evolutionary system E_{restr} by an evolutionary system $E = (\{0,1\}, K^{\{0,1\}})$ over the alphabet $\{0,1\}$. Thus, we show that there is an evolutionary system $E = (\Sigma, K)$ such that $\Rrightarrow^*_{\Sigma,K,r}$ is undecidable. The undecidability of $\Rrightarrow^*_{\Sigma,K}$ will be shown to be an immediate consequence of the undecidability of $\Rrightarrow^*_{\Sigma,K,r}$. This completes the proof of theorem 4.3, since the claim of this theorem is an immediate consequence of the undecidability $\Rrightarrow^*_{\Sigma,K}$.

Definition 15.2 *Let $D = \quad x_0 \Rrightarrow_r x_1 \Rrightarrow_r \cdots \Rrightarrow_r x_n$ be some derivation over the alphabet Σ. We say that D is <u>valid</u> in an evolutionary system $E = (\Sigma, K)$ if $\forall i \in 0..n, \forall w \in K : w \not\subseteq x_i$.*

We start with an injective mapping $\phi : \Sigma' \cup \{\$, \cent\} \to \{0,1\}^* \cup \{\$, \cent\}$. Let $\{\sigma_1, \ldots, \sigma_m\}$ be an enumeration of Σ'. Then we define

$$\begin{aligned} \phi(\sigma_i) &:= 11100^{i-1}10^{m-i}0111 \\ \phi(\$) &:= \$ \\ \phi(\cent) &:= \cent \end{aligned}$$

As usual, we use the symbol ϕ also for the homomorphic extension of ϕ to $(\Sigma' \cup \{\$, \cent\})^*$. Furthermore, we define a mapping γ of rules into words in $\{0,1\}^*$ by

$$\gamma(\{\sigma_i, \sigma_j\}) \quad := \quad \phi(\sigma_i) \vee \phi(\sigma_j),$$

where $\vee$ is the bitwise logical or.

The idea is to translate every word in $\$\Sigma'^*¢$ into $\$\{0,1\}^*¢$ using ϕ, and to simulate an application of a rule $\{\sigma_i, \sigma_j\}$ with $i < j$ by a sequence of substitutions

$$\begin{aligned} 11100^{i-1}10^{m-i}0111 &\Rrightarrow_r 11100^{i-1}10^{j-i-1}10^{m-j}0111 \\ &\Rrightarrow_r 11100^{i-j}10^{m-j}0111. \end{aligned} \tag{4}$$

Note that the middle word in (1) is just $\gamma(\{\sigma_i, \sigma_j\})$. Then for simulating E_{restr}, it is clear that $K^{\{0,1\}}$ must at least contain the translation of the lethal genes of E_{restr}, namely the set

$$\begin{aligned} K^{\{0,1\}}_{trans} &:= \{\phi(w) \mid w \in K_{con} \cup K_{inter}\} \\ &\cup \{\gamma(r) \mid r \in \mathcal{P}_2(\Sigma) \wedge r \notin R_S\}. \end{aligned}$$

Additionally, we need a set of lethal genes $K^{\{0,1\}}_{code}$ excluding all words that are not elements of

$$\begin{aligned} L_{code} &:= \$L^*_{trans}¢ \\ L_{trans} &:= \{w \in \{0,1\}^* \mid w \in \mathrm{img}(\phi) \cup \mathrm{img}(\gamma)\} \end{aligned}$$

But there is a problem, namely that only the simulation of substitutions uses intermediate words that are contained in L_{trans}. But this is not true of the simulation of insertions and deletions. The reason is just, that if $w \Rightarrow_{\{¢,\delta¢\}} w'$ with $w, w' \in \Sigma'^*$, then $|w|$ and $|w'|$ differ by one, but $|\phi(w)|$ and $|\phi(w')|$ differ by $m+8$. Hence, an insertion/deletion must be simulated by a sequence of rule applications in E inserting/deleting $m+8$ bits, thus producing some words that are not elements of L_{trans}. For this reason, we define

$$\begin{aligned} L_{code} &:= \$L^*_{trans}L_{pre}(\delta)¢ \\ L_{trans} &:= \{w \in \{0,1\}^* \mid w \in \mathrm{img}(\phi) \cup \mathrm{img}(\gamma)\} \\ L_{pre}(\delta) &:= \{w \mid w \text{ prefix of } \phi(\delta) \text{ and } w \neq \phi(\delta)\ \} \end{aligned}$$

Note that all words in L_{trans} have the same length, namely $m+8$.

In the following, we say that a word w is a <u>pre-codon</u> if $w = 0111$ or $w = \$$. These are the words directly before the encoding of a $\sigma \in \Sigma'$ in a word in L_{code}.

$$\begin{aligned} K^{\{0,1\}}_{code} := &\{w_p w \mid \quad w_p \text{ pre-codon } \wedge\ w \in \{0,1\}^* \\ &\qquad\qquad \wedge\ |w| = m+8\ \wedge\ w \notin L_{trans} \quad\} \qquad (5) \\ \cup\ &\{w_p w¢ \mid w_p \text{ pre-codon } \wedge\ |w| < m+8\ \wedge\ w \notin L_{pre}(\delta)\} \qquad (6) \end{aligned}$$

Note that words having more than six consecutive 1's contain an element of K_{code} as a convex subword.

Proposition 16.1 *Let $x \in \$\{0,1\}^*¢$. Then x contains a word in $K^{\{0,1\}}_{code}$ if and only if $x \notin L_{code}$.*

Proof. Let $x \in \$\{0,1\}^*\cent$. For the direction from left to right, let $x \in L_{code}$ and assume that

$$x = yw_pwz$$

such that $w_pw \in K_{code}^{\{0,1\}}$. Since $x \in L_{code}$, we know that

$$yw_p \in \$L_{trans}^* \text{ and } wz \in L_{trans}^* L_{pre}(\delta)\cent$$

If w_pw is contained in set defined in (5), then $|w| = m+8$. But then $w \in L_{trans}$, which is a contradiction. Similar, assuming w_pw is contained in set defined in (6) implies $z = \lambda$ since $x \in \$\{0,1\}^*\cent$. Hence, $w \in L_{pre}(\delta)\cent$, which is a contradiction.

For the direction from right to left, suppose $x \notin L_{code}$. Let $x = yz$ such that y is the longest prefix of x with $y \in \$L_{trans}^*$. Then there are w_p, w with

$$y = y'w_p \text{ and } z = wz'$$

such that w_p is a pre-codon, and $|w| = m+8$ or $|w| \leq m+8 \wedge z' = \lambda$. In both cases we get immediately $w_pw \in K_{code}^{\{0,1\}}$. □

Clearly, every derivation in E_{restr} can be simulated by a derivation in $E' = (\{0,1\}, K_{trans}^{\{0,1\}} \cup K_{code}^{\{0,1\}})$. For the back-translation, we have a problem similar to the one that occurred in the case of the simulation of a substitution system by a restricted substitution symbol, namely that we have to reorder the derivation in E for translating them back to derivations in E_{restr}. With the set of lethal genes $K_{trans}^{\{0,1\}} \cup K_{code}^{\{0,1\}}$, this is sometimes even impossible. Before we can present an example, we need some definitions. Given a word $w \in L_{code}$ and $i \in \mathbb{N}$, we define the i^{th} segment of w (written $\text{seg}_i(w)$) by

$$\text{seg}_i(w) \quad := \quad w_{i*(m+8)+2}^{(i+1)*(m+8)+1}$$

Note that $\text{seg}_i(w) = \lambda$ if $i*(m+8)+2 > |w|$. We say that a derivation

$$x_0 \Rrightarrow_r \ldots \Rrightarrow_r x_n$$

valid in $E' = (\{0,1\}, K_{trans}^{\{0,1\}} \cup K_{code}^{\{0,1\}})$ is E_{restr}-pure if for every $x_{i-1} \Rrightarrow_r x_i \Rrightarrow_r x_{i+1}$ we have

1. $x_i \in \text{img}(\phi)$, or
2. there is a j such that x_{i-1}, x_i and x_{i+1} differ only in the j^{th} segment.

Now let's turn to a concrete example showing that not every valid E'-derivation sequence can be reordered into an E_{restr}-pure sequence. Let

$$E_{restr} = (\{a,b\}, \delta, \{\{a,b\}\}, \{ab, ba\}) \text{ and } E' = (\{0,1\}, K_{trans}^{\{0,1\}} \cup K_{code}^{\{0,1\}})$$

Then

$$
\begin{aligned}
\phi(a) &= 1110100111\\
\phi(b) &= 1110010111\\
\gamma(\{a,b\}) &= 1110110111
\end{aligned}
$$

Now consider the derivation

$$
\begin{aligned}
\$\shortmid 1110\underline{10}0111\shortmid 1110\underline{10}0111\shortmid \text{¢} &\Rrightarrow_r \$\shortmid 1110\underline{11}0111\shortmid 1110\underline{10}0111\shortmid \text{¢}\\
&\Rrightarrow_r \$\shortmid 1110\underline{11}0111\shortmid 1110\underline{11}0111\shortmid \text{¢}\\
&\Rrightarrow_r \$\shortmid 1110\underline{01}0111\shortmid 1110\underline{11}0111\shortmid \text{¢}\\
&\Rrightarrow_r \$\shortmid 1110\underline{01}0111\shortmid 1110\underline{01}0111\shortmid \text{¢}
\end{aligned}
\tag{7}
$$

which is valid in E'. We have used $\shortmid$ to mark the boundaries of the different segments, and we have underlined the significant parts of sequences. For getting a E_{restr}-pure derivation, we would have to consider the derivation

$$
\begin{aligned}
\$\shortmid 1110\underline{10}0111\shortmid 1110\underline{10}0111\shortmid \text{¢} &\Rrightarrow_r \$\shortmid 1110\underline{11}0111\shortmid 1110\underline{10}0111\shortmid \text{¢}\\
&\Rrightarrow_r \$\shortmid 1110\underline{01}0111\shortmid 1110\underline{10}0111\shortmid \text{¢}\\
&\Rrightarrow_r \$\shortmid 1110\underline{01}0111\shortmid 1110\underline{11}0111\shortmid \text{¢}\\
&\Rrightarrow_r \$\shortmid 1110\underline{01}0111\shortmid 1110\underline{01}0111\shortmid \text{¢}.
\end{aligned}
\tag{8}
$$

which would translate into the $G_{E_{restr}}$-derivation

$$
\$aa\text{¢} \Rightarrow_{G_{E_{restr}}} \$ba\text{¢} \Rightarrow_{G_{E_{restr}}} \$bb\text{¢}
\tag{9}
$$

But (8) (resp. (9)) is not a valid E'-derivation (resp. E_{restr}-derivation), since for example the lethal gene $1110\underline{01}0111\shortmid 1110\underline{10}0111$ (resp. ba) is part of the second word. We can say, that the reason for this problem is that we started the simulation of two rule-applications in (7) in parallel, thus avoiding that the word ba becomes "visible". Hence, we have to avoid that two rule applications are simulated in parallel if their distance is smaller than the greatest lethal gene in $K^{\{0,1\}}_{trans}$. Hence, we define $K^{\{0,1\}}_{inter}$ as follows. Let $l_{\max}$ be $\max\{|w| \mid w \in K^{\{0,1\}}_{trans}\}$. Then

$$
\begin{aligned}
K^{\{0,1\}}_{inter} := {} & \{\gamma(r)w\gamma(r') \mid w \in L^*_{trans} \wedge r, r' \in R_S \wedge w \leq l_{\max}\}\\
\cup {} & \{\gamma(r)w\text{¢} \mid w \in L^*_{trans}L_{pre}(\delta) \wedge r \in R_S \wedge w \leq l_{\max}\}
\end{aligned}
$$

We define the evolutionary system E simulating E_{restr} by $E = (\{0,1\}, K^{\{0,1\}})$, where $K^{\{0,1\}} = K^{\{0,1\}}_{trans} \cup K^{\{0,1\}}_{code} \cup K^{\{0,1\}}_{inter}$.

Proposition 18.1 *Every derivation in E_{restr} can be simulated by a corresponding derivation in E.*

Proof. The simulation of substitution is done as described in (4). A rule application of the form $x\text{¢} \Rightarrow_{\{\text{¢},\delta\text{¢}\}} x\delta\text{¢}$ is simulated by $m + 8$ right insertion rules inserting successively the encoding $\phi(\delta)$ in front of ¢. By this method, it is guaranteed that all produced strings are elements of L_{code}. The deletion of one δ is simulated analogously. □

Proposition 18.2 *Given the valid E-derivation*

$$x \Rrightarrow_r y \Rrightarrow_r z$$

such that x, y *differ in the* i^{th} *segment of* x *and and* y, z *differ in the* j^{th} *segment with* $i \neq j \wedge |i - j| > \lfloor \frac{l_{\max}}{8+m} \rfloor$. *Then there is a* y' *such that*

$$x \ \Rightarrow_{\{\mathrm{seg}_j(x), \mathrm{seg}_j(z)\}} \ y' \ \Rightarrow_{\{\mathrm{seg}_i(x), \mathrm{seg}_i(z)\}} \ z$$

is a valid E-derivation.

Proof. We define

$$y' \ := \ \$\mathrm{seg}_0(x) \ldots \mathrm{seg}_{j-1}(x)\mathrm{seg}_j(z)\mathrm{seg}_{j+1}(x) \ldots$$

Clearly, we have both

$$x \ \Rightarrow_{\{\mathrm{seg}_j(x), \mathrm{seg}_j(z)\}} \ y' \ \Rightarrow_{\{\mathrm{seg}_i(x), \mathrm{seg}_i(z)\}} \ z$$

and

$$x \Rrightarrow_r y' \Rrightarrow_r z.$$

For proving that $x \Rrightarrow_r y' \Rrightarrow_r z$ is a valid E-derivation, we have only to show that no lethal gene in $K^{\{0,1\}}$ is a subword of y'. First, by definition of y' we have $y' \in L_{code}$, which implies that no element of $K^{\{0,1\}}_{code}$ is a subword of y' by Proposition 16.1. Furthermore, $|i - j| > \lfloor \frac{l_{\max}}{8+m} \rfloor$ implies that no element of $K^{\{0,1\}}_{inter}$ is a subword of y'.

The remaining part is to show that there is no $w \in K^{\{0,1\}}_{trans}$ such that $w \subseteq y'$. Suppose that such a w exists. If w is some $\gamma(r)$ with $r \notin R_S$, then there would be a k such that $w = \mathrm{seg}_k(y')$. By the definition of y', this would imply that either $\mathrm{seg}_k(x) = w$ or $\mathrm{seg}_k(z) = w$, which contradicts our assumption. If w is some $\phi(w')$ with $w' \in K_{con} \cup K_{inter}$, then there is some k and $l \leq \lfloor \frac{l_{\max}}{8+m} \rfloor$ such that

$$w = \mathrm{seg}_k(y') \ldots \mathrm{seg}_{k+l}(y').$$

If neither i, j is an element of the interval $[k, k+l]$, then

$$w = \mathrm{seg}_k(y') \ldots \mathrm{seg}_{k+l}(y') = \mathrm{seg}_k(x) \ldots \mathrm{seg}_{k+l}(x)$$

is a subword of x which contradicts the assumption that $x \Rrightarrow_r y \Rrightarrow_r z$ is a valid E-derivation. If $i \in [k, k+l]$, then $j \notin [k, k+l]$ since $|i - j| > \lfloor \frac{l_{\max}}{8+m} \rfloor \geq l$. This would again imply that w is a subword of x, which contradicts our assumption. If $j \in [k, k+l]$, then $i \notin [k, k+l]$ and therefore

$$w = \mathrm{seg}_k(y') \ldots \mathrm{seg}_{k+l}(y') = \mathrm{seg}_k(z) \ldots \mathrm{seg}_{k+l}(z).$$

But then w would be a subword of z, which contradicts the assumptions. Hence, for all $w \in K^{\{0,1\}}_{trans}$ we have $w \not\subseteq y'$. □

Lemma 19.1 *For every E-derivation D there is an equivalent E-derivation D' which is E_{restr}-pure. Furthermore, every E_{restr}-pure derivation in E can be translated into an equivalent E_{restr}-derivation.*

Theorem 19.2 *There is an evolutionary system $E = (\Sigma, K)$ such that the relations $\Rrightarrow^*_{\Sigma,K,r}$ and $\Rrightarrow^*_{\Sigma,K}$ are undecidable.*

Proof. The first claim follows from the undecidability of Thue-systems together with Lemmas 6.3, 15.1 and 19.1. The second claim follows from the fact that given a word $w \in L^*_{trans}$, then deleting symbol in w generates a word $w' \notin L_{trans}$. Since all words $w' \notin L_{trans}$ are excluded by $K^{\{0,1\}}_{code}$ according to Proposition 16.1, this implies that $\Rrightarrow^*_{\Sigma,K,r} = \Rrightarrow^*_{\Sigma,K}$. □

Theorem 4.3 is an immediate consequence of the last theorem.

3 Simulation of Turing Machine by Asexual Evolution

In this section, we show how to simulate an arbitrary 1-tape Turing machine M by a 1-tape *special* Turing machine M' (essentially a reversible Turing machine). ¿From this, it easily follows that M' can be simulated by a Thue system.

Definition 20.1 *A 1-tape Turing machine (abbreviated TM) is given by $M = (Q, \Gamma, \delta, s)$, where Q is a finite set of states containing the halt state h, Γ is a finite alphabet not containing the blank symbol $\#$, $s \in Q$ is the start state, and the transition function δ maps*

$$(Q - \{h\}) \times (\Gamma \times \{\#\}) \to Q \times (\Gamma \cup \{\#, L, R\}).$$

The TM M begins its computation on a 1-way infinite tape with input $w = w_1 \cdots w_n \in \Gamma^*$ in state s, with the tape head on the leftmost symbol.

$$\boxed{\#\quad w_1\quad w_2\quad \dots\quad w_n\quad \#}$$
$$\uparrow$$

A configuration α in the computation of TM M on input x is a word $[uqv]$, where $q \in Q$, $u, v \in (\Gamma \cup \{\#\})^*$, $|u|, |v| \geq 1$, the brackets ']', '[' do not belong to $\Gamma \cup \{\#\}$), and where $\#\#$ is neither a prefix of u nor suffix of v. The graph Δ of δ is the set

$$\{(q, a, p, b) : \delta(q, a) = (p, b)\}$$

As usual, if α, β are configurations of M, then $\alpha \vdash_M \beta$ means that β is the configuration immediately following α, using the transition function of M. As well, $\vdash^*_M$ is the reflexive, transitive closure of $\vdash_M$.

Let $M = (Q, \Sigma, \delta, s)$ be a 1-tape TM. Define the 1-tape TM $\widetilde{M}$ which has three tracks on its single tape, and begins its computation as follows:

$$\begin{array}{c}
\boxed{\begin{array}{ccccccc}
 & \# & w_1 & w_2 & \ldots & w_n & \# \\
 & \uparrow & & & & & \\
¢ & \$ & \$ & & & &
\end{array}} \\
\begin{array}{l}
\uparrow \\
q_0
\end{array}
\end{array}$$

The TM $\widetilde{M}$ simulates the movements of M and additionally keeps the history of the rules applied. Starting at the far left end of its tape, $\widetilde{M}$ sweeps to the right in a state similar to M's current state until the rightmost '\$' is encountered on the third track. In proceeding right, $\widetilde{M}$ determines what character M would have read, and performs the actions M would have on its second track. At the right end, $\widetilde{M}$ erases the '\$', writes the $r \in \Delta$ which M applied, and prints a '\$' to the right. Then $\widetilde{M}$ sweeps back to the leftmost '\$', and begins again. In summary, to simulate one of M's steps, $\widetilde{M}$ requires a sweep from the far left to far right, and back again. It follows that if M uses s tape cells and takes t steps in its computation on w, then $\widetilde{M}$ uses $3 + \max(s, t)$ tape cells and takes time $O(t^2)$.

In reading the transition rules of $\widetilde{\delta}$ below, it may help to know that state names have been chosen to indicate either current action or future action to be performed. For instance, $q_{j,R}^{r,R,\uparrow}$ is the state of $\widetilde{M}$ which simulates the state q_j of M, where the next action to be taken when in this state is to move right (indicated by superscript 'R'), then afterwards to write $\uparrow$, while $\widetilde{M}$ is simulating M's state q_j in its initial sweep to the right.

Formally, $\widetilde{M} = (\widetilde{Q}, \widetilde{\Sigma}, \widetilde{\delta}, s)$ where $\widetilde{Q}$ is

$$\{q_0, q_{i,R}, q_{i,R}^{r}, p_{i,R}^{r,L,\uparrow}, q_{i,R}^{r,R,\uparrow}, p_{i,R}^{r,\uparrow}, q_{i,R}^{r,\uparrow}, p_{i,R}^{r}, q_{i,R}^{r,\$}, p_{i,R}^{r,\$}, p_{i,R}^{\$,L}, q_{i,R}^{\$,L}, p_{i,L}^{L}, q_{i,L}^{L} : 0 \leq i \leq n, r \in \Delta\},$$

$\widetilde{\Sigma}$ is

$$(\Sigma \cup \{\#\}) \times \{\uparrow, \Box\} \times \{\$, \Box, ¢, r : r \in \Delta\},$$

and where $\widetilde{\delta}$ is described as follows.

1.	q_0	$\boxed{\begin{array}{c} \\ \\ \not{c}\end{array}}$	$q_{0,R}$	R	
2.	$q_{i,R}$	$\boxed{\begin{array}{c}\sigma\\ \neq\uparrow\\ \neq\\ \not{c}\end{array}}$	$q_{i,R}$	R	
3.	$q_{i,R}$	$\boxed{\begin{array}{c}\sigma\\ \uparrow\\ \gamma\end{array}}$	$q^r_{j,R}$	$\boxed{\begin{array}{c}\sigma'\\ \uparrow\\ \gamma\end{array}}$,	if $r = q_i\sigma q_j\sigma' \in \Delta$
4.	$q_{i,R}$	$\boxed{\begin{array}{c}\sigma\\ \uparrow\\ \gamma\end{array}}$	$p^{r,L,\uparrow}_{j,R}$	$\boxed{\begin{array}{c}\sigma\\ \\ \gamma\end{array}}$,	if $r = q_i\sigma q_jL \in \Delta$
5.	$q_{i,R}$	$\boxed{\begin{array}{c}\sigma\\ \uparrow\\ \gamma\end{array}}$	$q^{r,R,\uparrow}_{j,R}$	$\boxed{\begin{array}{c}\sigma\\ \\ \gamma\end{array}}$,	if $r = q_i\sigma q_jR \in \Delta$
6.	$p^{r,L,\uparrow}_{j,R}$	$\boxed{\begin{array}{c}\sigma\\ \\ \gamma\end{array}}$	$p^{r,\uparrow}_{j,R}$	L	
7.	$q^{r,R,\uparrow}_{j,R}$	$\boxed{\begin{array}{c}\sigma\\ \\ \gamma\end{array}}$	$q^{r,\uparrow}_{j,R}$	R	
8.	$p^{r,\uparrow}_{j,R}$	$\boxed{\begin{array}{c}\sigma\\ \\ \gamma\end{array}}$	$p^{r,R}_{j,R}$	$\boxed{\begin{array}{c}\sigma\\ \uparrow\\ \gamma\end{array}}$	

9.	$q_{j,R}^{r,\uparrow}$	$\boxed{\begin{array}{c}\sigma\\ \\ \gamma\end{array}}$	$q_{j,R}^{r}$	$\boxed{\begin{array}{c}\sigma\\ \uparrow\\ \gamma\end{array}}$
10.	$q_{j,R}^{r}$	$\boxed{\begin{array}{c}\sigma\\ \gamma\\ \neq \$\end{array}}$	$q_{j,R}^{r}$	R
11.	$q_{j,R}^{r}$	$\boxed{\begin{array}{c}\sigma\\ \gamma\\ \$\end{array}}$	$q_{j,R}^{r,\$}$	$\boxed{\begin{array}{c}\sigma\\ \gamma\\ r\end{array}}$
12.	$q_{j,R}^{r,\$}$	$\boxed{\begin{array}{c}\sigma\\ \gamma\\ r\end{array}}$	$q_{j,R}^{\$,L}$	R
13.	$q_{j,R}^{\$,L}$	$\boxed{\begin{array}{c}\sigma\\ \\ \end{array}}$	$q_{j,L}^{L}$	$\boxed{\begin{array}{c}\sigma\\ \\ \$\end{array}}$
14.	$q_{j,L}^{L}$	$\boxed{\begin{array}{c}\sigma\\ \\ \$\end{array}}$	$q_{j,L}$	L
15.	$q_{j,L}$	$\boxed{\begin{array}{c}\sigma\\ \gamma\\ \neq \$\end{array}}$	$q_{j,L}$	L
16.	$q_{j,L}$	$\boxed{\begin{array}{c}\sigma\\ \gamma\\ \$\end{array}}$	$q_{j,R}$	$\boxed{\begin{array}{c}\sigma\\ \gamma\\ \$\end{array}}$
17.	$p_{j,R}^{r,R}$	$\boxed{\begin{array}{c}\sigma\\ \uparrow\\ \gamma\end{array}}$	$p_{j,R}^{r}$	R
18.	$p_{j,R}^{r}$	$\boxed{\begin{array}{c}\sigma\\ \\ \neq \$\end{array}}$	$p_{j,R}^{r}$	R
19.	$p_{j,R}^{r}$	$\boxed{\begin{array}{c}\sigma\\ \\ \$\end{array}}$	$p_{j,R}^{r,\$}$	$\boxed{\begin{array}{c}\sigma\\ \gamma\\ r\end{array}}$
20.	$p_{j,R}^{r,\$}$	$\boxed{\begin{array}{c}\sigma\\ \gamma\\ r\end{array}}$	$p_{j,R}^{\$,L}$	R

21. $p^{\$,L}_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \\ \end{array}}$ $p^{L}_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \\ \$\end{array}}$

22. $p^{L}_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \\ \$\end{array}}$ $q_{j,L}$ L

We assume $\widetilde{M}$ starts on the following configuration.

$$\begin{array}{c} \boxed{\begin{array}{ccccccc} & \# & w_1 & w_2 & \dots & w_n & \# \\ & \uparrow & & & & & \\ ¢ & \$ & \$ & & & & \end{array}} \\ \begin{array}{ccccccc} \uparrow & & & & & & \\ q_0 & & & & & & \end{array} \end{array}$$

Definition 24.1 *If $M = (Q, \Gamma, \delta, q_0)$ is a TM, and $x \in \Gamma$, then $\alpha_{0,x}$ denotes the initial configuration of M on input x. The TM M is injective or 1-1 on $x \in \Sigma^*$ if for all $\alpha, \beta, \gamma \in \{\theta : \alpha_{0,x} \vdash_M^* \theta\}$ $\alpha \vdash_M \gamma$ and $\beta \vdash_M \gamma$ implies $\alpha = \beta$.*

The 1-tape TM M is special if it is 1-1 on all inputs x in Γ^* and for all configurations α, β, $\beta = (\beta_1, \dots, \beta_m)$ in $\{\theta : \alpha_{0,x} \vdash_M^* \theta\}$, if $\alpha \vdash_M \beta$, i is the position of M's head on β, and q is the state of M on β, then knowledge of $q, \beta_{i-1}, \beta_i, \beta_{i+1}$ uniquely determines the r of M used in the transition $\alpha \vdash_M \beta$.

Our notion of *special* 1-tape TM is a technical variant of Bennett's notion of *reversible* 3-tape TM.

Claim 24.2 *$\widetilde{M}$ is special.*

PROOF: The proof is by meticulous inspection of the transition rules of $\widetilde{\delta}$, using the encoded rule (e.g. rule r in $q^r_{j,R}$), or an adjacent symbol in β_{i-1}, β_i, β_{i+1} to disambiguate which rule was applied.

We summarize the considerations that must be made. Assume that a rule has just been performed, and that $\widetilde{M}$ is now in a new state q, and that β_{i-1}, β_i, β_{i+1} are the 3 symbols on $\widetilde{M}$'s tape in the immediate vicinity of $\widetilde{M}$'s head.

CASE 1: q is $q_{j,R}$.
Rule 1,2 or 16 must have been used. Rule 1 can be distinguished from the other two by whether β_{i-1} has ¢ on the third track. Rule 2 cannot be applied when the head is on the cell containing the rightmost '\$' on its third track; the head can be in this position only when applying rule 16.

CASE 2: q is $q^r_{j,R}$.
Rule 3,9 or 10 must have been used. Rules 3,9 can be distinguished by r. Note

that after application of rule 10, '$\uparrow$' cannot be in the middle track, since $q^r_{j,R}$ is a state entered only after M's rule has been successfully simulated. This distinguishes rule 10 from rules 3,9.

CASE 3: q is $p^r_{j,R}$.
Rule 17 or 18 must have been used. These rules can be distinguished by the presence of '$\uparrow$' on the second track in the vicinity.

CASE 4: q is $q_{j,L}$.
Rule 14,15 or 22 must have been used. These rules can be distinguished by looking at a history element (a rule) on the third track in the vicinity.

CASE 5: q is not of the above.
Then q is of the form $p^{r,L,\uparrow}_{j,R}$, $q^{r,R,\uparrow}_{j,R}$, $p^{r,\uparrow}_{j,R}$, $q^{r,\uparrow}_{j,R}$, $p^{r,R}_{j,R}$, $q^{r,\$}_{j,r}$, $q^{\$,L}_{j,R}$, $q^{L}_{j,L}$, $p^{r,\$}_{j,R}$, $p^{\$,L}_{j,R}$, or $p^L_{j,R}$. Each of these was obtained by a unique rule. ∎

Since $\widetilde{M}$ is special, one can simulate $\widetilde{M}$ in a straightforward manner by a Thue system, and hence by an evolutionary system. Since the special Turing machine $\widetilde{M}$ simulating 1-tape Turing machine M keeps the history of M's past, the evolutionary system E encodes M's past. In this sense, we have indicated a formal mechanism how the history of development of life could be encoded by individuals of a species.

4 Concluding remarks

In this paper, we have defined and analyzed a simple model for asexual evolution. Despite the extreme simplicity of the model, we have shown that one cannot answer questions such as whether given genomes x, y are in the same ergocity class modulo finite set K of lethal genes – i.e. whether there is a sequence of descendents from x to y.

We also wrote a simulation program in C to generate the set $N_k(x, K)$ of descendents (or neighbors) y within k generations of x, modulo a given set K of lethal genes. Our intent was to investigate the structure of the phylogeny tree below x, and to determine the relation between the Levenshtein distance from x to K (also the density or clustering of K) and the size of $N_k(x, K)$ as a function of k. This latter problem seems to be related to the earlier cited work of Chvátal-Sankoff [4], and appears to be quite difficult, as illustrated by the following example furnished by J. Sgall (personal communication). Let x be the empty word, $K_1 = \{01\}$ and $K_2 = \{00\}$. Then

$$\begin{aligned}
N_k(x, K_1) &= \{0^i, 1^i, 1^r 0^s \mid i \leq k, r+s \leq k\} \quad \text{and} \\
N_k(x, K_2) &= \{1^{n_1} 0 1^{n_2} 0 \cdots 0 1^{n_i} \mid n_j > 0 \text{ and } i - 1 + n_1 + \ldots n_i \leq k\} \\
&\cup \{0 1^{n_1} 0 1^{n_2} 0 \cdots 0 1^{n_i}, 1^{n_1} 0 1^{n_2} 0 \cdots 0 1^{n_i} 0 \mid n_j > 0 \text{ and } i + n_1 + \ldots n_i \leq k\} \\
&\cup \{0 1^{n_1} 0 1^{n_2} 0 \cdots 0 1^{n_i} 0 \mid n_j > 0 \text{ and } i + 1 + n_1 + \ldots n_i \leq k\}
\end{aligned}$$

It follows that the number of elements of $N_k(x, K_1)$ is linear in k, whereas that of $N_k(x, K_2)$ is exponential in k. However, under any sort of dissimilarity criterion, x is the same distance from K_1 as from K_2. As a curiosity, we include a sample phylogeny tree obtained by applying Felsenstein's PHYLIB package to the output of our program.

5 Acknowledgments

Special thanks go to Pavel Pudlák for several key discussions. We are indebted to him in particular for having explained his technique [11] for reducing to a rewrite system whose rules change at most one symbol, and for finding an error in a working draft of this paper. Thanks to K. Ramsey for several conversations and for suggesting a variant of the problem investigated in this article, to G. Mints for the references from S. Yu. Maslov, and D. Kerski who had suggested that undecidability results should be implicit in biological systems. Finally, thanks to Christa Mauer for some help with typing the manuscript.

References

[1] L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.

[2] S. Arora, Y. Rabani, and U. Vazirani. Simulating quadratic dynamical systems is $PSPACE$-complete (preliminary version). In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 459–467, 1994.

[3] C.H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18:766–776, 1989.

[4] V. Chvátal and D. Sankoff. Longest common subsequences of two random sequences. *Journal of Applied Probability*, 12:306–315, 1975.

[5] J. Dassow and V. Mitrana. Evolutionary grammars: a grammatical model for genome evolution. In *Computer Science and Biology*, pages pp. 87–92, 1996. Proceedings of the German Conference on Bioinformatics (GCB'96).

[6] L. Ilie and V. Mitrana. Crossing-over on languages: A formal representation of the recombination of genes in a chromosome. Technical Report 16, Turku Centre for Computer Science, June 1996.

[7] S. Yu. Maslov. The mutation calculi. *Journal of Soviet Mathematics*, 10:495–517, 1978.

[8] S. Yu. Maslov. Absorption relation on regular sets. *Journal of Soviet Mathematics*, 14:1468–1475, 1980.

[9] S. Yu. Maslov. *Theory of Deductive Systems and its Applications.* MIT Press (Cambridge), 1987.

[10] G. Paun and A. Salomaa. From DNA recombination to DNA computing, via formal languages. In *Computer Science and Biology*, pages pp. 93–98, 1996. Proceedings of the German Conference on Bioinformatics (GCB'96).

[11] P. Pudlák. Complexity theory and genetics. In *Proceedings of 9th Annual IEEE Conference on Structure in Complexity Theory*, 1994.

[12] M. S. Waterman. *Introduction to Computational Biology - Maps, sequences and genomes.* Chapman and Hall, 1995.

Figures 1 and 2 were produced with the alphabet A, C (rather than the full alphabet A, C, G, T), with starting genome $x = AACCC$, and with Lethal Gene set consisting of

ACCAA
CACCA
CCAAC
CACAA
CAACA
ACAAA

Since PHYLIB requires all strings to be of the same size, we generated those y from x such that $x \Rightarrow_{\{A,C\},K} y$ with 4 or fewer rule applications. Figure 1 is then the phylogeny tree output from PHYLIB for length 9 strings y, while Figure 2 for length 8 strings.

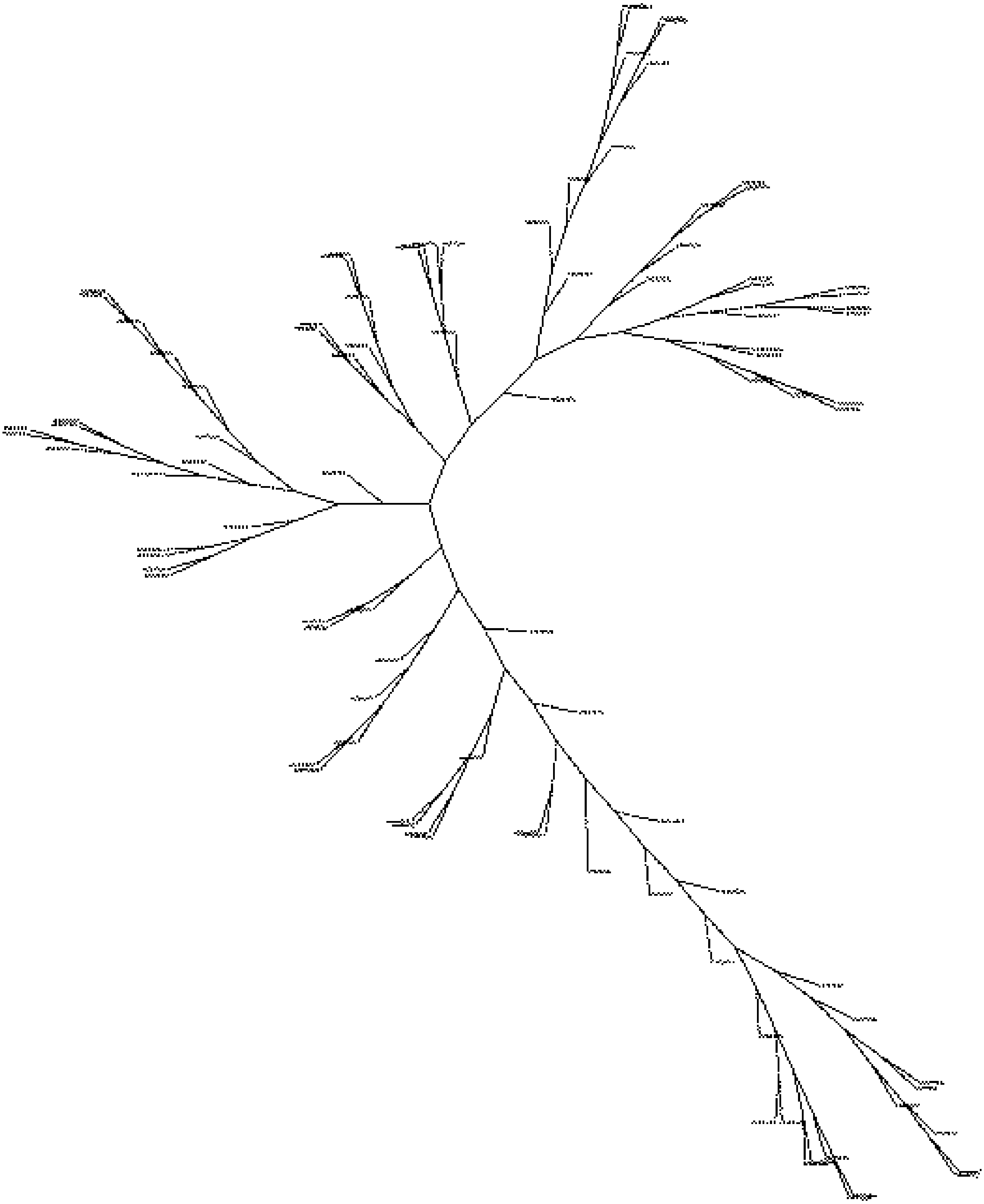

Figure 1:

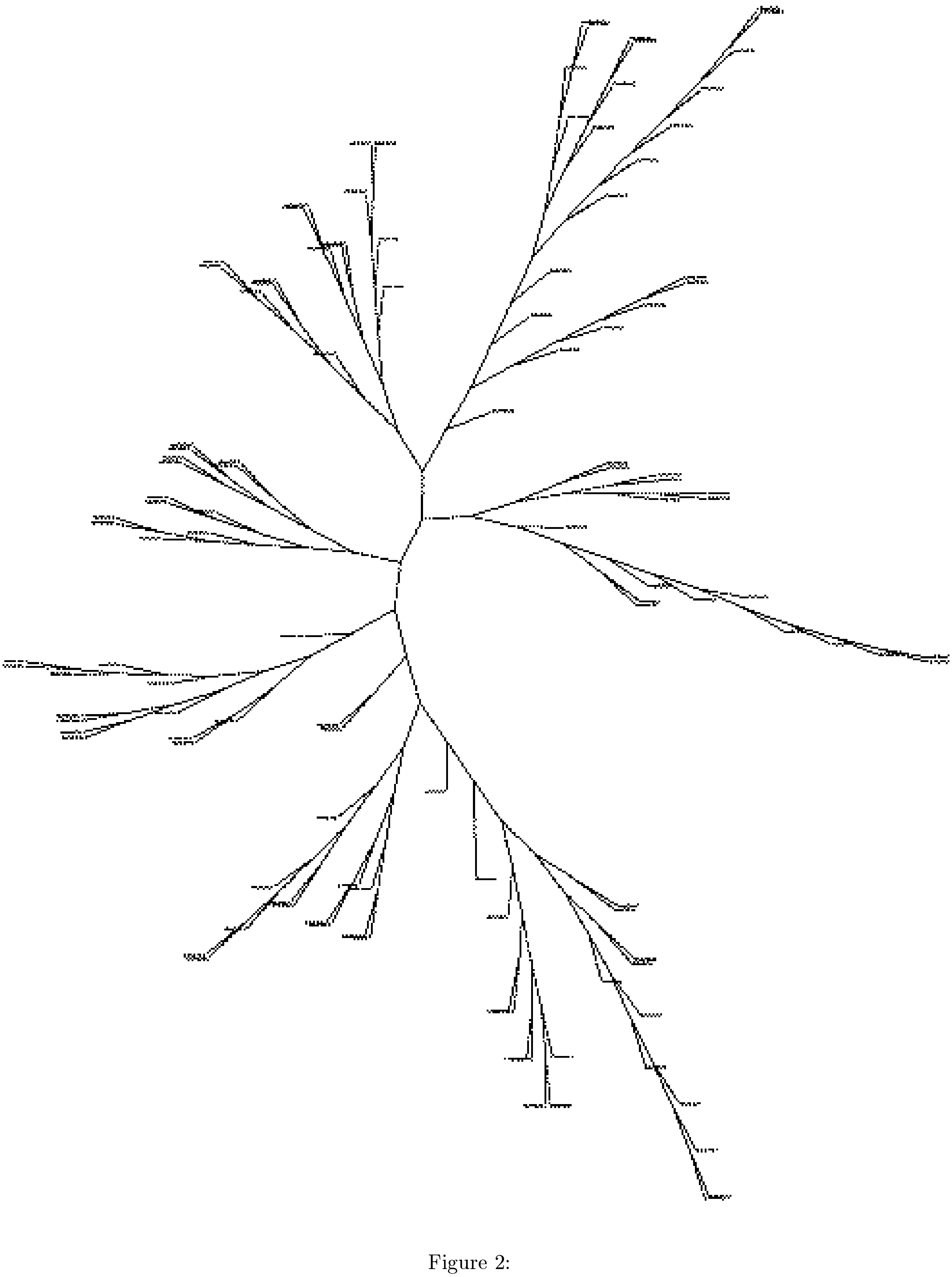

Figure 2: